1.1. Realiza un pequeño estudio de los principales lenguajes de programación y frameworks de Front-End y Back-End más solicitados en las ofertas de empleo en A Coruña y en España.

Java(cualquiera o 8 o +), Spring(general o Spring Boot), PHP, Laravel, Maven, BBDD relacionales(varios frameworks diferentes), Javascript (cualquier framework frontend),Github, HTML5, Scrum

1.2. Realiza un pequeño estudio de los principales IDE del mercado, específicos para lenguajes concretos o generalistas.

Para Python, los más usados son IDE Python sin PyCharm, Eclipse, AWS Cloud 9, The Jupyter Notebook y Spyder
De los más usados para Java son BlueJ, Eclipse, Visual Studio, y NETBEANS
Para múltiples lenguajes serían Eclipse, el Visual Studio y JetBrains

1.3. Realiza una sencilla aplicación de consola que tenga definida una clase llamada Persona con atributos privados: dni, nombre y edad. Añádele un constructor que incluya todos los atributos, getters, setters, toString, equals y hashcode basado en el dni.
Incluye un programa que defina un ArrayList con 6 personas (puedes meter sus valores por hardcode o hacer un sencillo método para que el usuario introduzca sus valores).
Desarrolla distintos métodos en el programa anterior con las siguientes características:
• Método al que se le pasa un ArrayList de Persona y devuelve la edad del mayor.
• Método al que se le pasa un ArrayList de Persona y devuelve la edad media.
• Método al que se le pasa un ArrayList de Persona y devuelve el nombre del mayor.
• Método al que se le pasa un ArrayList de Persona y devuelve la Persona mayor.
• Método al que se le pasa un ArrayList de Persona y devuelve todos los mayores de edad.
• Método al que se le pasa un ArrayList de Persona y devuelve todos los que tienen una edad
mayor o igual a la media.
En el main del programa haz llamadas a los métodos anteriores y muestra por pantalla su resultado.
Nota: a la hora de crear los métodos puedes reutilizar código de forma que unos llamen a otros y
minimizar el código duplicado.
Hay proyecto

1.4. Se desea hacer la gestión de las habitaciones de un hotel. Todas las habitaciones tienen un número
de habitación y un proceso de check-in y check-out. En el hotel hay tres tipos de habitaciones, aunque
podría haber más en el futuro:
- 3 habitaciones Lowcost (cuesta 50 euros/día todo el año).
- 10 habitaciones dobles. Tienen una tarifa normal de 100 euros/día y un incremento del 20%
si el día de salida es abril, julio o agosto.
- 5 habitaciones Suite. 200 euros/día con 20% de descuento para estancias de 10 o más días.
• Debes crear una o más clases para las habitaciones y una clase para el Hotel. La clase Hotel tendrá las 18 habitaciones en un ArrayList y las marcará inicialmente como "no ocupadas".
• El programa tendrá un menú para hacer check-in entre las habitaciones libres, check-out entre las ocupadas y listar habitaciones libres y ocupadas.
• El check-in es común para todas las habitaciones, consiste en marcar la habitación como ocupada. El check-out consiste en marcar la habitación como libre y calcular el importe a pagar que se calculará en función del tipo de habitación y de los días de estancia (quizás sea necesario almacenar la fecha de llegada en el momento del check-in)
• Sugerencia: Para probar el programa, al hacer el check-out, puedes considerar cada día como un segundo, así, si han pasado 3 segundos, considerar 3 días.
Cuestión 1: ¿Habitación debería ser una clase abstracta o una interfaz? ¿Por qué?
Clase abstracta. Todas los tipos de habitación tienen que ser habitaciones. No es que les diga que tienen que tener unos métodos. **Són** habitaciones, así que tienen que extender la clase madre habitación, no implementarla como interfaz.
Cuestión 2: ¿Es útil que el método toString () en la clase Habitación?
Si quieres, por ejemplo, mostrar la habitación de los clientes antes de realizar el pago por si hay algún error, mucho.

Hay proyecto

1.5. Se desea desarrollar un programa gestione los dispositivos domóticos de un edificio. Para ello tendremos un ArrayList que contenga en principio 3 elementos, uno para el termostato de la calefacción, otro para el ascensor y otro para el dial de la radio del hilo musical, pero en el futuro podríamos tener más elementos. El termostato tiene una fecha de última revisión, un valor entero en grados centígrados: mínimo 15, máximo 80 y la temperatura inicial es 20. El ascensor tiene una planta en la que se encuentra, pudiendo ser desde 0 a 8. La planta inicial es la cero. El dial de radio va desde 88.0 a 104.0 avanzando de décima en décima, siendo el valor inicial 88.0. De cada elemento (y los futuros que aparezcan) deben ser capaces de realizar las siguientes funciones:

• subir(), incrementa una unidad el elemento domótico. Devuelve true si la operación se realiza correctamente o false si no se puede hacer por estar al máximo.

• bajar(): decrementa una unidad el elemento domótico. Devuelve true si la operación se realiza correctamente o false si no se puede hacer por estar al mínimo.

• reset(): pone en la situación inicial el elemento domótico. No devuelve nada.

• verEstado(): Devuelve un String con el tipo de dispositivo y su estado actual.

Además, el termostato debe incluir un nuevo método:

• revisar(). Fija la fecha de revisión a la fecha actual. No devuelve nada. Una vez definido el sistema, crea un programa que inicie un ArrayList con una instancia de cada uno de los 3 dispositivos y luego mediante un menú nos permita hacer operaciones, primero qué operación queremos realizar(

0:Salir

1:subir un dispositivo

2:bajar un dispositivo

3: resetear un dispositivo,

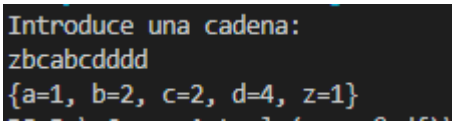
4:revisar termostato) y luego seleccionar sobre que elemento queremos trabajar (verificando que sea un valor entre 0 y el tamaño del ArrayList -1)

• El menú, además de las opciones, nos mostrará siempre el estado de todos los dispositivos.

Hay proyecto

1.6. Realizar una clase llamada Primitiva que tenga definido una colección de 6 elementos con el resultado de un sorteo de la primitiva (serán 6 enteros con valores comprendidos entre 1 y 49 y sin repetidos). Los números se deberán mostrar ordenados ascendentemente así que decide cual es la colección que mejor se adapta a estos requisitos. La clase dispondrá de un constructor en el que se generan y almacenan esos números al azar, también tendrá un método al que se le pase una combinación jugada como parámetro (no necesariamente ordenada) y devuelva el número de aciertos. Realiza a continuación un programa en el que el usuario introduzca boletos (6 números sin repetidos) y le diga cuantos acertó. Realizar control de errores, tanto si el usuario introduce valores no numéricos, números repetidos o valores no comprendidos entre 1 y 49.
Hay proyecto

1.7. Realizar un programa donde el usuario introduce un String y se muestre la cantidad de veces que aparece cada letra (ordenadas alfabéticamente, no por orden de aparición). Para tener un rendimiento óptimo, se debe recorrer el String solo una vez. Elige la colección óptima para minimizar el código necesario. Ejemplo:



Hay proyecto

1.8. Realiza un programa de consola para la gestión de los empleados de una empresa sobre una base de datos SQLite. De los empleados mantenemos un identificador único para cada empleado, su nombre completo y su salario. El programa inicialmente cargará la tabla de empleados desde un fichero csv y luego presentará un menú para realizar las siguientes operaciones:

• Alta: solicita el id, nombre y salario. Opcional: puedes validar que no exista el id.

• Baja: solicita un id y si lo encuentra elimina el empleado correspondiente.

• Modificación: solicita un id y si lo encuentra, solicita nuevo nombre y salario y lo modifica.

• Consulta: solicita salario mínimo y máximo, y muestra los empleados con salario comprendido entre los valores introducidos.

El menú, además de las acciones anteriores, siempre mostrará el conjunto de empleados existentes en la tabla en cada momento.

El profesor te entregará un esqueleto con el código del programa (carpeta CrudEmpleadoJdbc) para que lo completes y también el archivo con la base de datos SQLite, aunque este último se puede crear desde el plugin de Chrome: 'Sqlite Manager'. En la siguiente imagen se ve el proceso: se crearía la tabla con el nombre que queramos (por ejemplo: empleados) y luego en el botón inferior "Save" guardamos la base de datos, que contiene esa única tabla, con el nombre que queramos. La base de datos se guardará en un solo fichero.

Cuestiones:

• Estudia qué es el patrón de diseño "Repository" y comprueba si en este ejercicio se cumple.

• ¿Qué ventajas aporta el uso de una clase BdManagerImpl y una interfaz BdManager?

Hay proyecto